Internet Engineering Task Force (IETF)
Request for Comments: 8582
Category: Standards Track
ISSN: 2070-1721
August 2019

S. Donovan, Ed.
Oracle
E. Noel
AT&T Labs

Diameter Overload Rate Control

Abstract

This specification documents an extension to the Diameter Overload Indication Conveyance (DOIC) base solution, which is defined in RFC 7683. This extension adds a new overload-control abatement algorithm. This abatement algorithm allows for a DOIC reporting node to specify a maximum rate at which a DOIC reacting node sends Diameter requests to the DOIC reporting node.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a new Diameter overload-control abatement algorithm, the "rate" algorithm.

The base Diameter overload specification [RFC7683] defines the "loss algorithm" as the default Diameter overload loss abatement algorithm. The loss algorithm allows a reporting node (see Section 3) to instruct a reacting node (see Section 3) to reduce the amount of traffic sent to the reporting node by abating (diverting or throttling) a percentage of requests sent to the server. While this can effectively decrease the load handled by the server, it does not directly address cases where the rate of arrival of service requests changes quickly. For instance, if the service requests that result in Diameter transactions increase quickly, then the loss algorithm cannot guarantee the load presented to the server remains below a specific rate level. The loss algorithm can be slow to ensure the stability of reporting nodes when subjected to rapidly-changing loads. The "loss" algorithm errs both in throttling too much when there is a dip in offered load, and throttling not enough when there is a spike in offered load.

Consider the case where a reacting node is handling 100 service requests per second, where each of these service requests results in one Diameter transaction being sent to a reporting node. If the reporting node is approaching an overload state, or is already in an overload state, it will send a Diameter Overload report requesting a percentage reduction in traffic sent when the loss algorithm is used as a Diameter overload abatement algorithm. Assume for this discussion that the reporting node requests a 10% reduction. The reacting node will then abate (diverting or throttling) ten Diameter transactions a second, sending the remaining 90 transactions per second to the reporting node.

Now assume that the reacting node's service requests spike to 1000 requests per second. The reacting node will continue to honor the reporting node's request for a 10% reduction in traffic. This results, in this example, in the reacting node sending 900 Diameter transactions per second, abating the remaining 100 transactions per second. This spike in traffic is significantly higher than the reporting node is expecting to handle and can result in negative impacts to the stability of the reporting node.

The reporting node can, and likely would, send another Overload report requesting that the reacting node abate 91% of requests to get back to the desired 90 transactions per second. However, once the spike has abated and the rate at which the reacting node handles requests has returned to 100 per second, this will result in just 9 transactions per second being sent to the reporting node, requiring a new Overload report setting the reduction percentage back to 10%. This control feedback loop has the potential to make the situation worse by causing wide fluctuations in traffic on multiple nodes in the Diameter network.

One of the benefits of a rate-based algorithm over the loss algorithm is that it better handles spikes in traffic. Instead of sending a request to reduce traffic by a percentage, the rate approach allows the reporting node to specify the maximum number of Diameter requests per second that can be sent to the reporting node. For instance, in this example, the reporting node could send a rate-based request specifying the maximum transactions per second to be 90. The reacting node will send the 90 regardless of whether it is receiving 100 or 1000 service requests per second.

It should be noted that one of the implications of the rate-based algorithm is that the reporting node needs to determine how it wants to distribute its load over the set of reacting nodes from which it is receiving traffic. For instance, if the reporting node is receiving Diameter traffic from 10 reacting nodes and has a capacity of 100 transactions per second, then the reporting node could choose to set the rate for each of the reacting nodes to 10 transactions per second. This, of course, is assuming that each of the reacting nodes has equal performance characteristics. The reporting node could also choose to have a high-capacity reacting node send 55 transactions per second and the remaining 9 low-capacity reacting nodes send 5 transactions per second. The ability of the reporting node to specify the amount of traffic on a per-reacting-node basis implies that the reporting node must maintain state for each of the reacting nodes. This state includes the current allocation of Diameter traffic to that reacting node. If the number of reacting nodes changes, either because new nodes are added, nodes are removed from service, or nodes fail, then the reporting node will need to redistribute the maximum Diameter transactions over the new set of reacting nodes.

This document extends the base Diameter Overload Indication Conveyance (DOIC) solution [RFC7683] to add support for the rate abatement algorithm.

This document draws heavily on work in the SIP Overload Control Working Group. The definition of the rate abatement algorithm is copied almost verbatim from the SIP Overload Control (SOC) document [RFC7415], with changes focused on making the wording consistent with the DOIC solution and the Diameter protocol.

2. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminology

Diameter Node

A Diameter Client, Diameter Server, or Diameter Agent [RFC6733]

Diameter Endpoint

A Diameter Client or Diameter Server [RFC6733]

DOIC Node

A Diameter node that supports the DOIC solution defined in [RFC7683]

Reporting Node

A DOIC node that sends an Overload report in a Diameter answer message

Reacting Node

A DOIC node that receives and acts on a DOIC Overload report

4. Interaction with DOIC Report Types

As of the publication of this specification, there are three DOIC report types:

HOST_REPORT 0:
Overload of a specific Diameter application at a specific Diameter node as defined in [RFC7683]

REALM_REPORT 1:
Overload of a specific Diameter application at a specific Diameter realm as defined in [RFC7683]

PEER_REPORT 2:
Overload of a specific Diameter peer as defined in [RFC8581]

The rate algorithm MAY be selected by reporting nodes for any of these report types.

It is expected that all report types defined in the future will indicate whether or not the rate algorithm can be used with that report type.

5. Capability Announcement

This document defines the rate abatement algorithm (referred to as "rate" in this document) feature. Support for the rate feature by a DOIC node will be indicated by a new value of the OC-Feature-Vector attribute-value pair (AVP), as described in Section 7.1.1, per the rules defined in [RFC7683].

Since all nodes that support DOIC are required to support the loss algorithm, DOIC nodes supporting the rate feature will support both the loss and rate abatement algorithms.

DOIC reacting nodes supporting the rate feature MUST indicate support for both the loss and rate algorithms in the OC-Feature-Vector AVP and MAY indicate support for other algorithms.

As defined in [RFC7683], a DOIC reporting node supporting the rate feature selects a single abatement algorithm in the OC-Feature-Vector AVP and OC-Peer-Algo AVP in the answer message sent to the DOIC reacting nodes.

A reporting node can select one abatement algorithm to apply to Host and Realm reports, and a different algorithm to apply to peer reports.

For Host or Realm reports, the selected algorithm is reflected in the OC-Feature-Vector AVP sent as part of the OC-Supported-Features AVP included in answer messages for transactions where the request contained an OC-Supported-Features AVP. This is per the procedures defined in [RFC7683].

For Peer reports, the selected algorithm is reflected in the OC-Peer-Algo AVP sent as part of the OC-Supported-Features AVP included in answer messages for transactions where the request contained an OC-Supported-Features AVP. This is per the procedures defined in [RFC8581].

6. Overload-Report Handling

This section describes any changes to the behavior defined in [RFC7683] for the handling of Overload reports when the rate abatement algorithm is used.

6.1. Reporting-Node OCS

A reporting node that uses the rate abatement algorithm SHOULD maintain reporting-node Overload Control State (OCS) for each reacting node to which it sends a rate Overload Report (OLR).

Note: This is different from the behavior defined in [RFC7683] where a reporting node sends a single loss percentage to all reacting nodes.

A reporting node SHOULD maintain OCS entries when using the rate abatement algorithm per supported Diameter application, per targeted reacting node and per report type.

A rate OCS entry is identified by the tuple of Application-ID, report type, and DiameterIdentity of the target of the rate OLR.

The rate OCS entry SHOULD include the rate allocated to the reacting node.

A reporting node that has selected the rate abatement algorithm MUST indicate the rate requested to be applied by DOIC reacting nodes in the OC-Maximum-Rate AVP included in the OC-OLR AVP.

All other elements for the OCS defined in [RFC7683] and [RFC8581] also apply to the reporting node’s OCS when using the rate abatement algorithm.

6.2. Reacting-Node OCS

A reacting node that supports the rate abatement algorithm MUST indicate rate as the selected abatement algorithm in the reacting-node OCS based on the OC-Feature-Vector AVP or the OC-Peer-Algo AVP in the received OC-Supported-Features AVP.

A reacting node that supports the rate abatement algorithm MUST include the rate specified in the OC-Maximum-Rate AVP included in the OC-OLR AVP as an element of the abatement-algorithm-specific portion of reacting-node OCS entries.

All other elements for the OCS defined in [RFC7683] and [RFC8581] also apply to the reporting nodes OCS when using the rate abatement algorithm.

6.3. Reporting-Node Maintenance of OCS

A reporting node that has selected the rate abatement algorithm and enters an overload condition MUST indicate rate as the abatement algorithm and MUST indicate the selected rate in the resulting reporting-node OCS entries.

When selecting the rate algorithm in the response to a request that contained an OC-Supporting-Features AVP with an OC-Feature-Vector AVP indicating support for the rate feature, a reporting node MUST ensure that a reporting-node OCS entry exists for the target of the Overload report. The target is defined as follows:

- o For Host reports, the target is the DiameterIdentity contained in the Origin-Host AVP received in the request.
- o For Realm reports, the target is the DiameterIdentity contained in the Origin-Realm AVP received in the request.
- o For Peer reports, the target is the DiameterIdentity of the Diameter peer from which the request was received.

A reporting node that receives a capability announcement from a new reacting node, meaning a reacting node for which it does not have an OCS entry, and the reporting node that chooses the rate algorithm for that reacting node may need to recalculate the rate to be allocated to all reacting nodes. Any changed rate values will be communicated in the next OLR sent to each reacting node.

6.4. Reacting-Node Maintenance of OCS

When receiving an answer message indicating that the reporting node has selected the rate algorithm, a reacting node MUST indicate the rate abatement algorithm in the reacting-node OCS entry for the reporting node.

A reacting node receiving an Overload report for the rate abatement algorithm MUST save the rate received in the OC-Maximum-Rate AVP contained in the OC-OLR AVP in the reacting-node OCS entry.

6.5. Reporting-Node Behavior for Rate Abatement Algorithm

When in an overload condition with rate selected as the overload abatement algorithm and when handling a request that contained an OC-Supported-Features AVP that indicated support for the rate abatement algorithm, a reporting node SHOULD include an OC-OLR AVP for the rate algorithm using the parameters stored in the reporting-node OCS for the target of the Overload report.

> Note: It is also possible for the reporting node to send Overload reports with the rate algorithm indicated even when the reporting node is not in an overloaded state. This could be a strategy to proactively avoid entering into an overloaded state. Whether or not to do so is up to local policy.

When sending an Overload report for the rate algorithm, the OC-Maximum-Rate AVP MUST be included in the OC-OLR AVP and the OC-Reduction-Percentage AVP MUST NOT be included.

6.6. Reacting-Node Behavior for Rate Abatement Algorithm

When determining if abatement treatment should be applied to a request being sent to a reporting node that has selected the rate abatement algorithm, the reacting node can choose to use the algorithm detailed in Section 8.

Other algorithms for controlling the rate MAY be implemented by the reacting node. Any algorithm implemented MUST correctly limit the maximum rate of traffic being sent to the reporting node.

Once a determination is made by the reacting node that an individual Diameter request is to be subjected to abatement treatment, then the procedures for throttling and diversion defined in [RFC7683] and [RFC8581] apply.

7. Rate Abatement Algorithm AVPs

7.1. OC-Supported-Features AVP

The rate algorithm does not add any new AVPs to the OC-Supported-Features AVP.

The rate algorithm does add a new feature bit to be carried in the OC-Feature-Vector AVP.

7.1.1. OC-Feature-Vector AVP

This extension adds the following capability to the OC-Feature-Vector AVP.

OLR_RATE_ALGORITHM (0x0000000000000004)

This bit is assigned to the rate abatement algorithm. When this flag is set by the overload-control endpoint, it indicates that the DOIC node supports the rate abatement algorithm.

7.2. OC-OLR AVP

This extension defines the OC-Maximum-Rate AVP to be an optional part of the OC-OLR AVP.

```
OC-OLR ::= < AVP Header: 623 >
           < OC-Sequence-Number >
           < OC-Report-Type >
           [ OC-Reduction-Percentage ]
           [ OC-Validity-Duration ]
           [ SourceID ]
           [ OC-Maximum-Rate ]
         * [ AVP ]
```

This extension makes no changes to the other AVPs that are part of the OC-OLR AVP.

This extension does not define new Overload report types. The existing report types of HOST_REPORT and REALM_REPORT defined in [RFC7683] apply to the rate control algorithm. The report type of PEER_REPORT defined in [RFC8581] also applies to the rate control algorithm.

7.2.1. OC-Maximum-Rate AVP

The OC-Maximum-Rate AVP (AVP code 670) is of type Unsigned32 and describes the maximum rate that the sender is requested to send traffic. This is specified in terms of requests per second.

A value of zero indicates that no traffic is to be sent.

7.3. Attribute-Value Pair Flag Rules

Attribute Name	AVP Code	Section Defined	Value Type	AVP flag rules: MUST	AVP flag rules: MUST NOT
OC-Maximum-Rate	670	7.2.1	Unsigned32		V

8. Rate Abatement Algorithm

This section is pulled from [RFC7415] with minor changes needed to make it apply to the Diameter protocol.

8.1. Overview

The reporting node is the one protected by the overload control algorithm defined here. The reacting node is the one that abates traffic towards the server.

Following the procedures defined in [RFC7683], the reacting node and reporting node signal their support for rate-based overload control.

Then, periodically, the reporting node relies on internal measurements (e.g., CPU utilization or queuing delay) to evaluate its overload state and estimate a target maximum Diameter request rate in number of requests per second (as opposed to target percent reduction in the case of loss-based abatement).

When in an overloaded state, the reporting node uses the OC-OLR AVP to inform reacting nodes of its overload state and of the target Diameter request rate.

Upon receiving the Overload report with a target maximum Diameter request rate, each reacting node applies overload abatement for new Diameter requests towards the reporting node.

8.2. Reporting-Node Behavior

The actual algorithm used by the reporting node to determine its overload state and estimate a target maximum Diameter request rate is beyond the scope of this document.

However, the reporting node MUST periodically evaluate its overload state and estimate a target Diameter request rate beyond which it would become overloaded. The reporting node must allocate a portion of the target Diameter request rate to each of its reacting nodes. The reporting node may set the same rate for every reacting node, or may set different rates for different reacting nodes.

The maximum rate determined by the reporting node for a reacting node applies to the entire stream of Diameter requests, even though abatement may only affect a particular subset of the requests, since the reacting node might apply priority as part of its decision of which requests to abate.

When setting the maximum rate for a particular reacting node, the reporting node may need to take into account the workload (e.g., CPU load per request) of the distribution of message types from that reacting node. Furthermore, because the reacting node may prioritize the specific types of messages it sends while under overload restriction, this distribution of message types may be different from the message distribution for that reacting node under non-overload conditions (e.g., either higher or lower CPU load).

Note that the value of OC-Maximum-Rate AVP (in request messages per second) for the rate algorithm provides a loose upper bound on the traffic sent by the reacting node to the reporting node.

In other words, when multiple reacting nodes are being controlled by an overloaded reporting node, at any given time, some reporting nodes may receive requests at a rate below its target maximum Diameter request rate while receiving others above that target rate. But, the resulting request rate presented to the overloaded reporting node will converge towards the target Diameter request rate or a lower rate.

Upon detection of overload, and the determination to invoke overload controls, the reporting node follows the specifications in [RFC7683] to notify its reacting nodes of the allocated target maximum Diameter request rate, and to notify them that the rate abatement is in effect.

The reporting node uses the OC-Maximum-Rate AVP defined in this specification to communicate a target maximum Diameter request rate to each of its clients.

8.3. Reacting-Node Behavior

8.3.1. Default Algorithm for Rate-Based Control

A reference algorithm is shown below.

Note that use of "//" below indicates a comment.

No priority case:

```
     // T: inter-transmission interval, set to 1 / OC-Maximum-Rate
     // TAU: tolerance parameter
     // ta: arrival time of the most recent arrival
     // LCT: arrival time of last Diameter request that
     //      was sent to the server
     //      (initialized to the first arrival time)
     // X: current value of the leaky bucket counter (initialized to
     //    TAU0)

     // After most recent arrival, calculate auxiliary variable Xp
     Xp = X - (ta - LCT);

     if (Xp <= TAU) {
       // Transmit Diameter request
       // Update X and LCT
       X = max (0, Xp) + T;
       LCT = ta;
     } else {
       // Reject Diameter request
       // Do not update X and LCT
     }
```

In determining whether or not to transmit a specific message, the reacting node can use any algorithm that limits the message rate to the OC-Maximum-Rate AVP value in units of messages per second. For ease of discussion, we define T = 1/[OC-Maximum-Rate] as the target inter-Diameter request interval. It may be strictly deterministic, or it may be probabilistic. It may or may not have a tolerance factor, to allow for short bursts, as long as the long-term rate remains below 1/T.

The algorithm may have provisions for prioritizing traffic.

If the algorithm requires other parameters (in addition to "T", which is 1/OC-Maximum-Rate), they may be set autonomously by the reacting node, or they may be negotiated independently between the reacting node and the reporting node.

In either case, the coordination is out of the scope of this document. The default algorithms presented here (one with and one without provisions for prioritizing traffic) are only examples.

To apply abatement treatment to new Diameter requests at the rate specified in the OC-Maximum-Rate AVP value sent by the reporting node to its reacting nodes, the reacting node MAY use the proposed default algorithm for rate-based control or any other equivalent algorithm that forward messages in conformance with the upper bound of 1/T messages per second.

The default leaky bucket algorithm presented here is based on Appendix A.2 of [ITU-T-I.371]. The algorithm makes it possible for reacting nodes to deliver Diameter requests at a rate specified in the OC-Maximum-Rate value with tolerance parameter TAU (preferably configurable).

Conceptually, the leaky bucket algorithm can be viewed as a finite capacity bucket whose real-valued content drains out at a continuous rate of 1 unit of content per time unit and whose content increases by the increment T for each forwarded Diameter request. T is computed as the inverse of the rate specified in the OC-Maximum-Rate AVP value, namely T = 1 / OC-Maximum-Rate.

Note that when the OC-Maximum-Rate value is 0 with a non-zero OC-Validity-Duration, then the reacting node should apply abatement treatment to 100% of Diameter requests destined to the overloaded reporting node. However, when the OC-Validity-Duration value is 0, the reacting node should stop applying abatement treatment.

If, at a new Diameter request arrival, the content of the bucket is less than or equal to the limit value TAU, then the Diameter request is forwarded to the server; otherwise, the abatement treatment is applied to the Diameter request.

Note that the capacity of the bucket (the upper bound of the counter) is (T + TAU).

The tolerance parameter TAU determines how close the long-term admitted rate is to an ideal control that would admit all Diameter requests for arrival rates less than 1/T and then admit Diameter requests precisely at the rate of 1/T for arrival rates above 1/T. In particular, at mean arrival rates close to 1/T, it determines the tolerance to deviation of the inter-arrival time from T. (The larger TAU, the more tolerance to deviations from the inter-departure interval T.)

This deviation from the inter-departure interval influences the admitted rate burstiness or the number of consecutive Diameter requests forwarded to the reporting node (burst size proportional to TAU over the difference between 1/T and the arrival rate).

In situations where reacting nodes are configured with some knowledge about the reporting node and other traffic sources (e.g., operator pre-provisioning), it can be beneficial to choose a value of TAU based on how many reacting nodes will be sending requests to the reporting node.

Reporting nodes with a very large number of reacting nodes, each with a relatively small arrival rate, will generally benefit from a smaller value for TAU in order to limit queuing (and hence response times) at the reporting node when subjected to a sudden surge of traffic from all reacting nodes. Conversely, a reporting node with a relatively small number of reacting nodes, each with a proportionally larger arrival rate, will benefit from a larger value of TAU.

Once the control has been activated, at the arrival time of the k-th new Diameter request, ta(k), the content of the bucket is provisionally updated to the value

X' = X - (ta(k) - LCT)

where X is the value of the leaky bucket counter after arrival of the last forwarded Diameter request, and LCT is the time at which the last Diameter request was forwarded.

If X' is less than or equal to the limit value TAU, then the new Diameter request is forwarded and the leaky bucket counter X is set to X' (or to 0 if X' is negative) plus the increment T, and LCT is set to the current time ta(k). If X' is greater than the limit value TAU, then the abatement treatment is applied to the new Diameter request, and the values of X and LCT are unchanged.

When the first response from the reporting node has been received, indicating control activation (OC-Validity-Duration>0), LCT is set to the time of activation, and the leaky bucket counter is initialized to the parameter TAU0 (preferably configurable), which is 0 or larger but less than or equal to TAU.

TAU can assume any positive real number value and is not necessarily bounded by T.

TAU=4*T is a reasonable compromise between burst size and abatement rate adaptation at low offered rate.

Note that specification of a value for TAU, and any communication or coordination between servers, is beyond the scope of this document.

8.3.2. Priority Treatment

A reference algorithm is shown below.

Priority case:

```
  // T: inter-transmission interval, set to 1 / OC-Maximum-Rate
  // TAU1: tolerance parameter of no priority Diameter requests
  // TAU2: tolerance parameter of priority Diameter requests
  // ta: arrival time of the most recent arrival
  // LCT: arrival time of last Diameter request that
  //      was sent to the server
  //      (initialized to the first arrival time)
  // X: current value of the leaky bucket counter (initialized to
  //    TAU0)

  // After most recent arrival, calculate auxiliary variable Xp
  Xp = X - (ta - LCT);

 if (AnyRequestReceived && Xp <= TAU1) || (PriorityRequestReceived &&
  Xp <= TAU2 && Xp > TAU1) {
    // Transmit Diameter request
    // Update X and LCT
    X = max (0, Xp) + T;
    LCT = ta;
  } else {
    // Apply abatement treatment to Diameter request
    // Do not update X and LCT
  }
```

The reacting node is responsible for applying message priority and for maintaining two categories of requests: request candidates for reduction, and requests not subject to reduction (except under extenuating circumstances when there aren't any messages in the first category that can be reduced).

Accordingly, the proposed leaky bucket implementation is modified to support priority using two thresholds for Diameter requests in the set of request candidates for reduction. With two priorities, the proposed leaky bucket requires two thresholds TAU1 < TAU2:

- o All new requests would be admitted when the leaky bucket counter is at or below TAU1.

- o Only higher priority requests would be admitted when the leaky bucket counter is between TAU1 and TAU2.

- o All requests would be rejected when the bucket counter is above TAU2.

This can be generalized to n priorities using n thresholds for n>2.

With a priority scheme that relies on two tolerance parameters (TAU2 influences the priority traffic, and TAU1 influences the non-priority traffic), always set TAU1 <= TAU2 (TAU is replaced by TAU1 and TAU2). Setting both tolerance parameters to the same value is equivalent to having no priority. TAU1 influences the admitted rate the same way as TAU does when no priority is set, and the larger the difference between TAU1 and TAU2, the closer the control is to strict priority queuing.

TAU1 and TAU2 can assume any positive real number value and is not necessarily bounded by T.

Reasonable values for TAU0, TAU1, and TAU2 are:

- o TAU0 = 0,

- o TAU1 = 1/2 * TAU2, and

- o TAU2 = 10 * T.

Note that specification of a value for TAU1 and TAU2, and any communication or coordination between servers, is beyond the scope of this document.

8.3.3. Optional Enhancement: Avoidance of Resonance

As the number of reacting-node sources of traffic increases and the throughput of the reporting node decreases, the maximum rate admitted by each reacting node needs to decrease, and therefore the value of T becomes larger. Under some circumstances, e.g., if the traffic arises very quickly simultaneously at many sources, the occupancies of each bucket can become synchronized, resulting in both the admissions from each source being close in time and batched, or very "peaky" arrivals at the reporting node. This gives rise not only to control instability, but also very poor delays and even lost messages. An appropriate term for this is "resonance" [Erramilli].

If the network topology is such that resonance can occur, then a simple way to avoid resonance is to randomize the bucket occupancy at two appropriate points: at the activation of control, and whenever the bucket empties, as described below:

After updating the value of the leaky bucket to X', generate a value u as follows:

if X' > 0, then u=0

else, if X' <= 0, then let u be set to a random value uniformly distributed between -1/2 and +1/2

Then, (only) if the arrival is admitted, increase the bucket content by an amount T + uT, which will therefore be just T if the bucket hadn't emptied, or lie between T/2 and 3T/2 if it had.

This randomization should also be done when control is activated, i.e., instead of simply initializing the leaky bucket counter to TAU0, initialize it to TAU0 + uT, where u is uniformly distributed as above. Since activation would have been a result of the response to a request sent by the reacting node, the second term in this expression can be interpreted as being the bucket increment following that admission.

This method has the following characteristics:

- If TAU0 is chosen to be equal to TAU and all sources activate control at the same time due to an extremely high request rate, then the time until the first request admitted by each reacting node would be uniformly distributed over [0,T];
- The maximum occupancy is TAU + (3/2)T, rather than TAU + T without randomization;
- For the special case of "classic gapping", where TAU=0, then the minimum time between admissions is uniformly distributed over [T/2, 3T/2], and the mean time between admissions is the same, i.e., T+1/R where R is the request arrival rate.
- At high load, randomization rarely occurs. Therefore, there is no loss of precision of the admitted rate, even though the randomized "phasing" of the buckets remains.

9. IANA Considerations

IANA has registered the following values in the "Authentication, Authorization, and Accounting (AAA) Parameters" registry:

One new AVP code is defined in Section 7.2.1.

One new OC-Feature-Vector AVP value is defined in Section 7.1.1.

9.1. OC-Supported-Features

As indicated in Section 7.1.1, a new allocation has been made for the OC-Feature-Vector AVP.

10. Security Considerations

The rate abatement mechanism is an extension to the base Diameter Overload mechanism. As such, all of the security considerations outlined in [RFC7683] apply to the rate abatement mechanism.

In addition, the rate algorithm could be used to handle denial-of-service (DoS) attacks more effectively than the loss algorithm.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <https://www.rfc-editor.org/info/rfc6733>.

[RFC7683] Korhonen, J., Ed., Donovan, S., Ed., Campbell, B., and L. Morand, "Diameter Overload Indication Conveyance", RFC 7683, DOI 10.17487/RFC7683, October 2015, <https://www.rfc-editor.org/info/rfc7683>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8581] Donovan, S., "Diameter Agent Overload and the Peer Overload Report", RFC 8581, DOI 10.17487/RFC8581, August 2019, <https://www.rfc-editor.org/info/rfc8581>.

11.2. Informative References

[Erramilli]
Erramilli, A. and L. Forys, "Traffic Synchronization Effects In Teletraffic Systems", 1991.

[ITU-T-I.371]
ITU-T, "Traffic control and congestion control in B-ISDN", ITU-T Recommendation I.371, March 2004.

[RFC7415] Noel, E. and P. Williams, "Session Initiation Protocol (SIP) Rate Control", RFC 7415, DOI 10.17487/RFC7415, February 2015, <https://www.rfc-editor.org/info/rfc7415>.

Acknowledgements

The authors would like to thank Lionel Morand for his contributions to this document.

Authors' Addresses

Steve Donovan (editor)
Oracle
7460 Warren Parkway, Suite 300
Frisco, Texas 75034
United States of America

Email: srdonovan@usdonovans.com

Eric Noel
AT&T Labs
200s Laurel Avenue
Middletown, NJ 07747
United States of America

Email: ecnoel@research.att.com